



LTE-xxx 4G网关DTU Modbus转JSON 用户手册

众联万物 智慧未来

我们用心创造

目录

一、 功能简介.....	1
1.1 功能特点.....	1
1.2 网关工作流程.....	2
三、 使用说明.....	4
3.1 阿里云 IoT 平台接入.....	4
3.2 自建服务器-MQTT 协议.....	14
3.3 自建服务器-TCP 协议.....	18
四、 Lua 代码说明.....	20
五、 NTP 时间和 GPS 数据获取.....	23
5.1 NTP 时间获取.....	23
5.2 GPS 数据采集.....	24
六、 自定义 json 模板.....	23
七、 其他特殊关键字(服务器下发 json 指令).....	29
7.1 SHELL.....	29
7.2 luaCode.....	30
7.3 Reg1-Reg10.....	32

一、功能简介

Modbus 转 json 功能主要用于采集 Modbus RTU 传感器数据，将采集的数据按配置的数据类型解析，并且打包成 key-value 的 json 格式上报到服务器，支持服务器下发 json 格式命令，省去了用户自己解析数据和转换格式的麻烦。支持连接自建服务器和阿里云 IoT 平台。

1.1 功能特点

- 支持连接自建服务器和阿里云 IoT 平台
- 支持 TCP-Client/UDP-Master/TCP-ZSD/UDP-ZSD/MQTT/HTTP 协议
- 支持 Modbus RTU 转 json，支持采集多个寄存器数据打包成 json 上传到服务器
- 支持 json 转 Modbus，支持接收服务器 json 指令，转换为 Modbus RTU 输出
- 支持自定义 json 格式
- 支持 NTP 时间
- 支持本地、远程参数配置
- 支持公式运算，可以将原始数据按照一定公式转换为需要的数据
- 支持 Lua 代码，用户可以对采集的数据做判断，并且执行不同的操作
- 支持关键字下发 json 指令配置脚本，lua 代码，Reg 寄存器的值

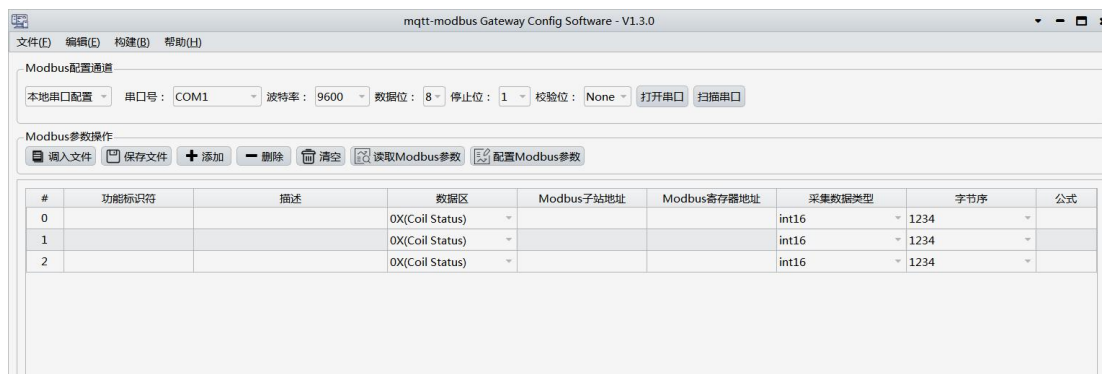
1.2 网关工作流程

json 网关 DTU 通过配置 Modbus 相关参数往串口发送相应的 Modbus 指令，485 设备收到指令以后会返回采集的数据，网关 DTU 通过数据类型进行解析，并将解析以后的数值以 key-value 的格式打包成 json 发送到服务器。流程如下：



- 1) 配置服务器 IP 地址/域名和端口号，如果是 MQTT 协议还需要配置 ClientID、username、password、发布主题和订阅主题等参数，配置轮询周期和上报周期、json 数据格式。
- 2) 配置 Modbus 相关参数，Modbus 子站地址、寄存器地址、数据类型、计算公式。
- 3) 如果是阿里云 IoT 平台，需要在服务器上创建设备；如果是自建服务器，需要在服务器端打开相应的上位机软件。
- 4) 服务器接收到打包的 json 数据。

二、参数说明



参数项	说明
功能标识符	用户自定义，只能是数字或者英文字符
描述	对标签的描述
数据区	Modbus RTU 功能码：0X-01, 1X-02, 3X-04, 4X-03
Modbus 子站地址	Modbus RTU 子站地址
Modbus 寄存器地址	Modbus RTU 寄存器地址
采集数据类型	共 7 种数据类型，上传的报文根据数据类型进行解析 int16: 两字节，有符号 uint16: 两字节，无符号 int32: 四字节，有符号 uint32: 四字节，无符号 float: 单精度浮点数，四字节 bool: bool 类型值，只有 0,1 两种类型，一般用于开关量 BCD-2: 两字节 BCD 码 BCD-4: 四字节 BCD 码 BCD-8: 八字节 BCD 码
字节序	int32/uint32/float 类型数据才有字节序，其他类型此参数无效
公式	采集数据的计算公式，格式为 $valuey = valuex * a + b$ ， $valuex$ 为原始值， $valuey$ 为经过公式计算后的值，上报的数据为 $valuey$ ，如果此项为空则取原始值上报。 注：bool 类型不支持公式运算
保存文件	保存 json 配置文件
调入文件	调入保存的 json 配置文件
添加	添加功能标识符（采集指令）
删除	删除功能标识符（采集指令）
清空	清空所有配置的参数
读取 Modbus 参数	读取配置的 Modbus 参数
配置 Modbus 参数	配置 Modbus 参数

三、使用说明

3.1 阿里云 IoT 平台接入

3.1.1 进入阿里云 IoT 平台官网 <https://www.aliyun.com>，登录账号，然后点击右上角的控制台。

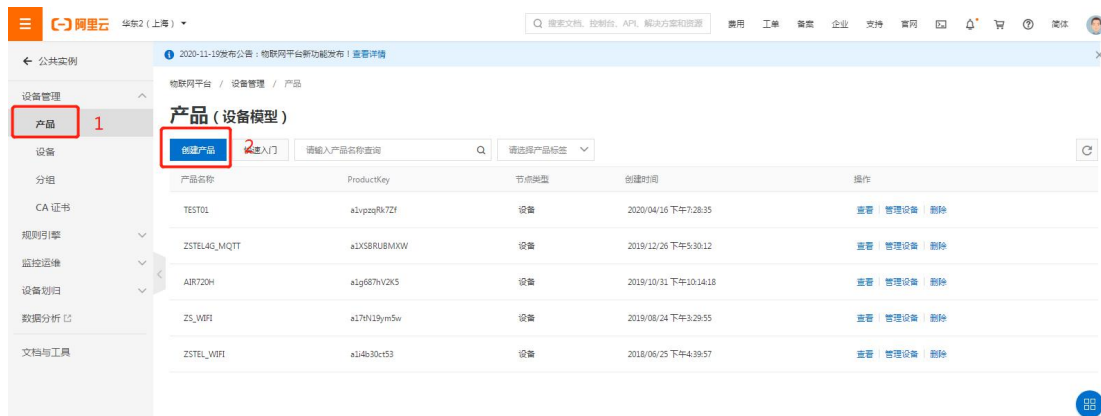


3.1.2 进入控制台后选择产品与服务-物联网平台。

注：第一次使用阿里云 IoT 平台物联网服务器需要先开通服务才能使用。



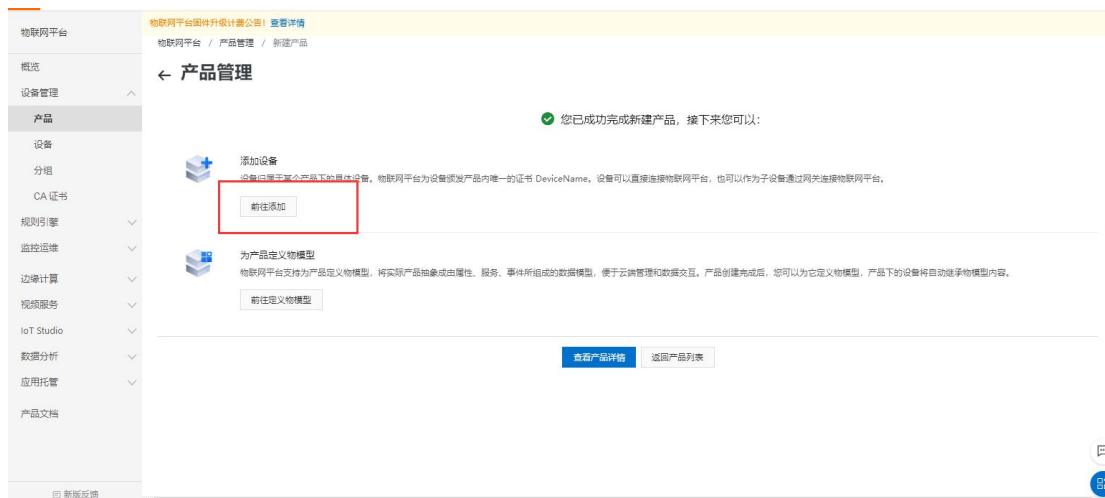
3.1.3 选择产品-创建产品。

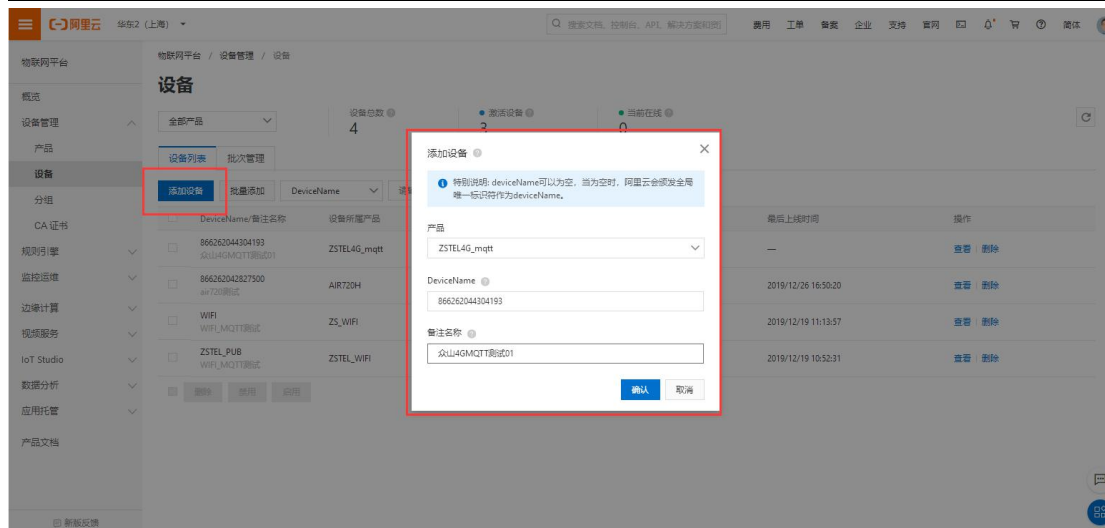


3.1.4 创建产品，产品名称可以自己定义，其他参数按图片上的配置即可。

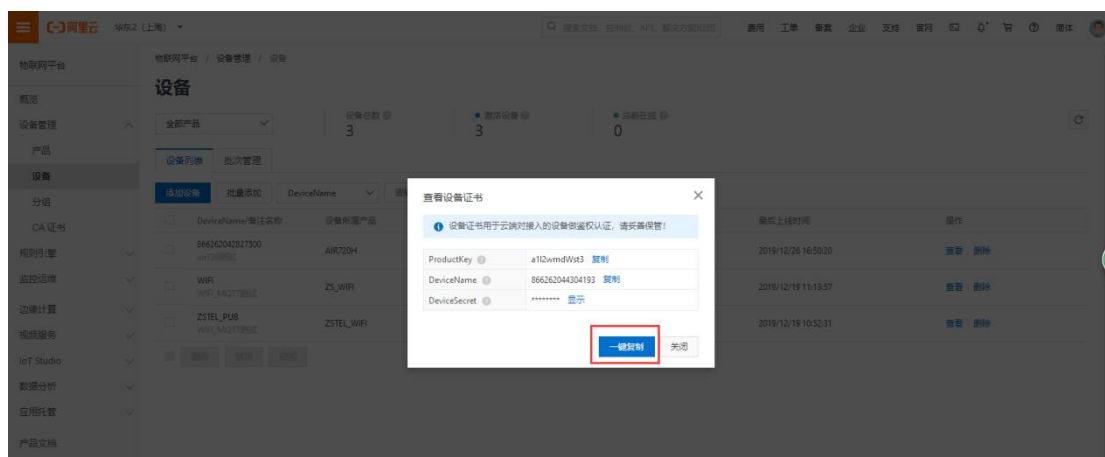


3.1.5 创建产品成功以后添加设备。



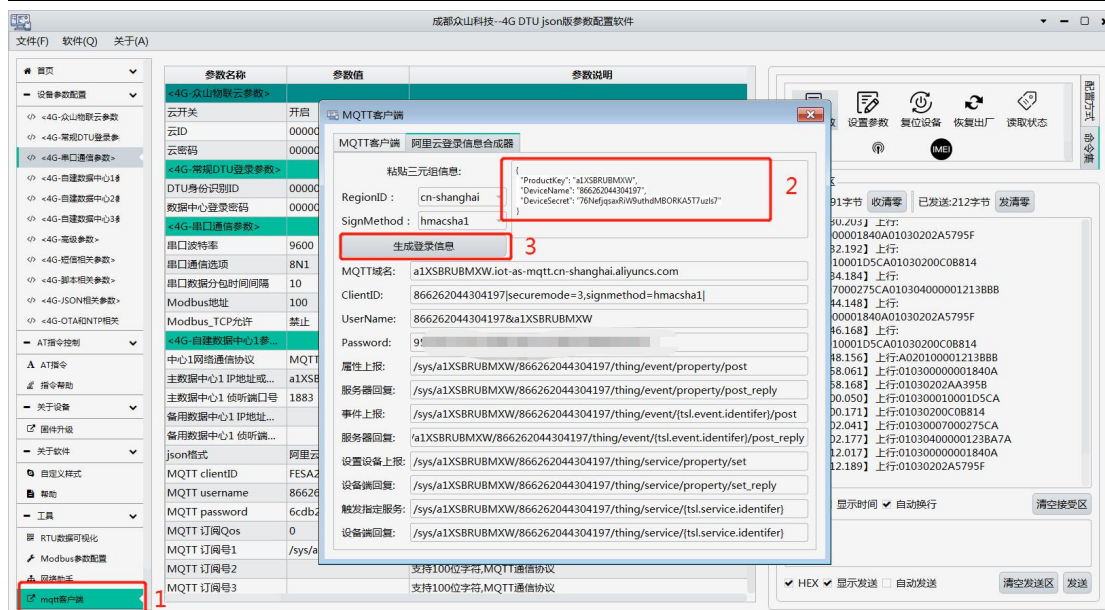


3.1.6 添加好设备后会弹出设备鉴权证书, 新建一个文本文档, 点击一键复制, 将信息复制到一个 TXT 文档里。



3.1.7 配置连接阿里云 IoT 平台的相关参数。

打开参数配置软件左下角的 MQTT 客户端, 选择阿里云登录信息合成器, 将信息复制到粘贴区, 点击生成登录信息, 将相关信息填入参数配置软件对应的位置。



打开“DTUCFG-V1.1.5”，将生成的阿里云 IoT 平台 MQTT 相关参数配置进对应的参数项。发布号配置“属性上报”内容

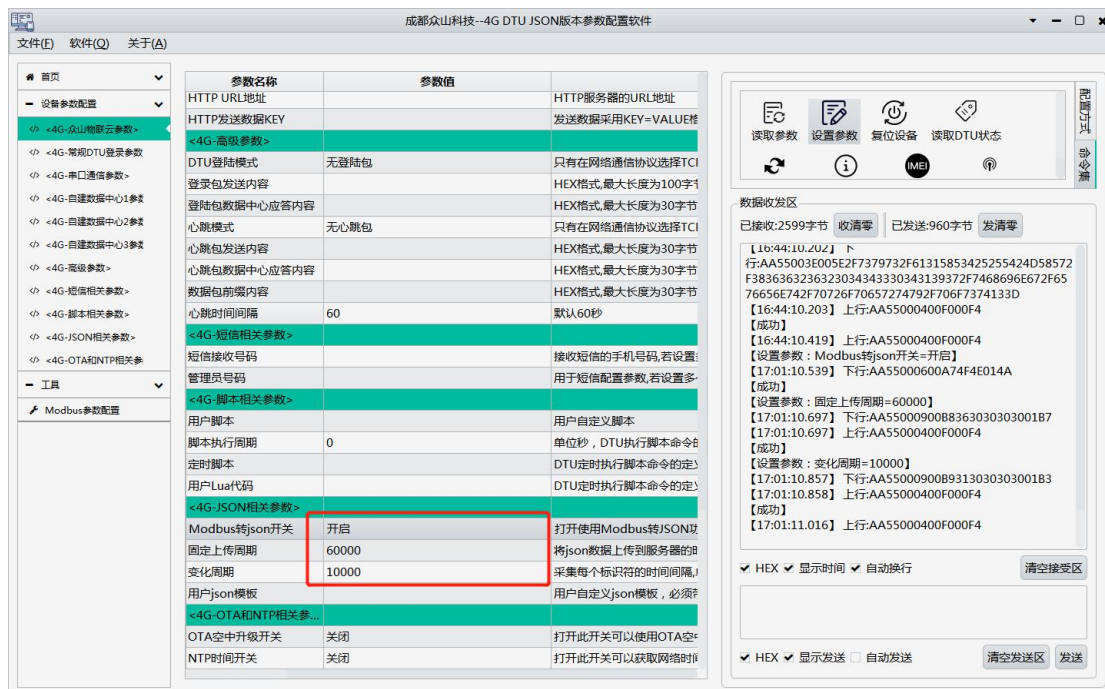
/sys/a1X5BRUBMXW/866262044304197/thing/event/property/post

订阅号配置“设置设备属性”内容：

/sys/a1X5BRUBMXW/866262044304197/thing/service/property/set

阿里云 IoT 平台相关参数配置完以后需要配置“变化周期”和“固定上传周期”，变化周期表示发送 Modbus 指令的时间间隔，固定上传周期表示将 json 包发送到服务器的周期，单位都为毫秒。

注：除了连接阿里云 IoT 平台的必要参数，还需要将 Modbus 转 json 开关打开，json 格式设置为阿里云 IoT 平台。



3.1.8 切换到 Modbus RTU 参数, 根据实际需要采集的 Modbus RTU 设备配置相应的参数, 本手册用一个温湿度-光照度传感器示例, Modbus RTU 地址及寄存器定义如下:

4.4.2 读取设备地址 0x01 的温湿度值

问询帧

地址码	功能码	起始地址	数据长度	校验码低位	校验码高位
0x01	0x03	0x00,0x00	0x00,0x02	0xC4	0x0B

应答帧（例如读到温度-10.1℃，湿度 65.8%RH）

地址码	功能码	有效字数	湿度值	温度值	校验码低位	校验码高位
0x01	0x03	0x04	0x02 0x92	0xFF 0x9B	0x5A	0x3D

温度：

当温度低于 0℃ 时以补码形式上传

FF9B H(十六进制)= -101 =>温度= -10.1℃

湿度：

292 H(十六进制)=658=>湿度= 65.8%RH

4.4.3 读取设备地址 0x01 的光照度值

问询帧

地址码	功能码	起始地址	数据长度	校验码低	校验码高位
0x01	0x03	0x00 0x07	0x00 0x02	0x75	0xCA

应答帧（例如读到光照度为 132854Lux）

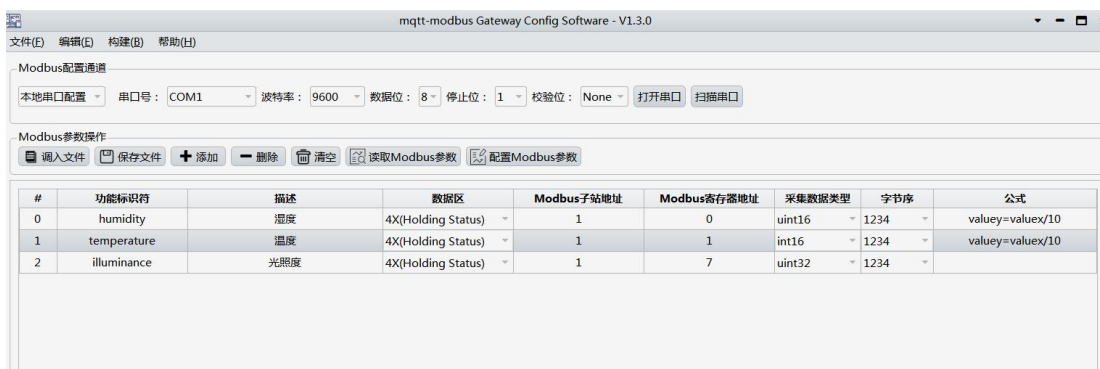
地址码	功能码	返回有效字节数	数据区	校验码低位	校验码高位
0x01	0x03	0x04	0x00 0x02 0x06 0xF6	0xD8	0x15

18

可以看出温度和湿度为两个字节，温度有符号，数据类型是 int16；湿度无符号，数据类型是 uint16，转换为十进制以后都需要除以 10，公式配置为 $valuey = valutex / 10$ 。光照度为四个字节，无符号，数据类型为 uint32，字节序为 1234，无公式，取原始值。参数配置如下：

点击工具-Modbus 参数配置可以进入 Modbus 配置界面

注：点击添加按钮添加功能标识符，删除按钮删除功能标识符。



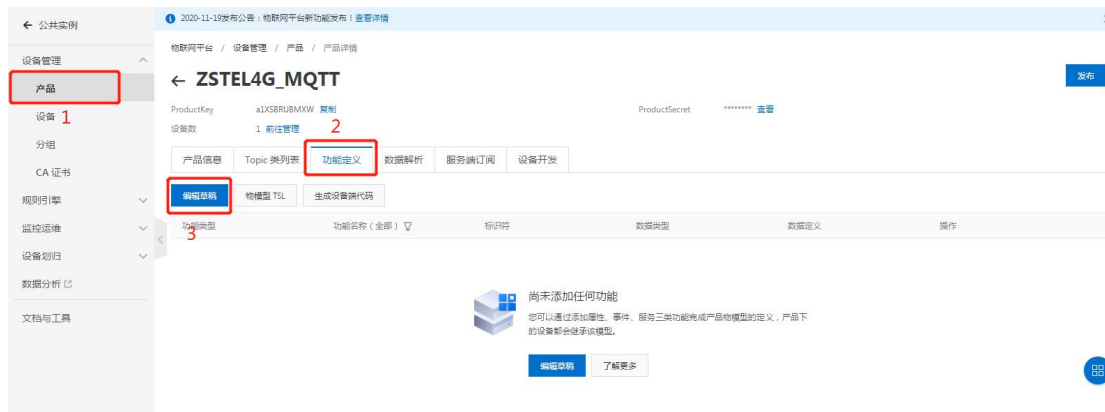
参数配置成功以后需要点复位设备，参数才会生效

3.1.9 阿里云 IoT 平台物理模型配置

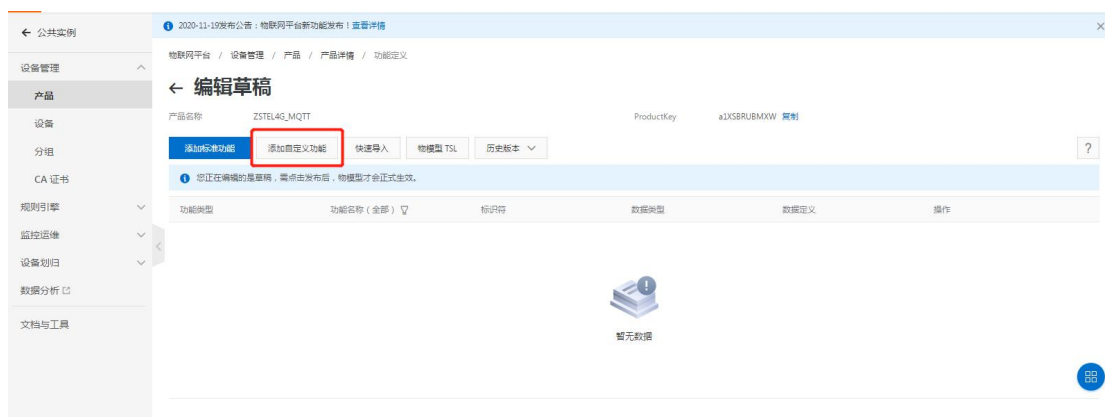
在阿里云 IoT 平台的产品-功能定义里面设置每个功能标识符的定义（根据每个寄存器的实际类型定义，不然阿里云 IoT 平台平台会提示参数类型错误），定义好以后点发布。当 DTU 采集数据上报后可以在设备-物理模型里面看数据，数据是 Modbus RTU 协议解析后的数据。

注：虽然实际温度是 int16 类型，湿度是 uint16 类型，但经过公式计算以后有小数，所以阿里云 IoT 平台平台上定义温度和湿度的数据类型为 float 浮点数类型才能正确解析；光照度不需要通过公式计算，上报的值为原始值，类型为 uint32。

(1) 点击产品-功能定义，选择编辑草稿。

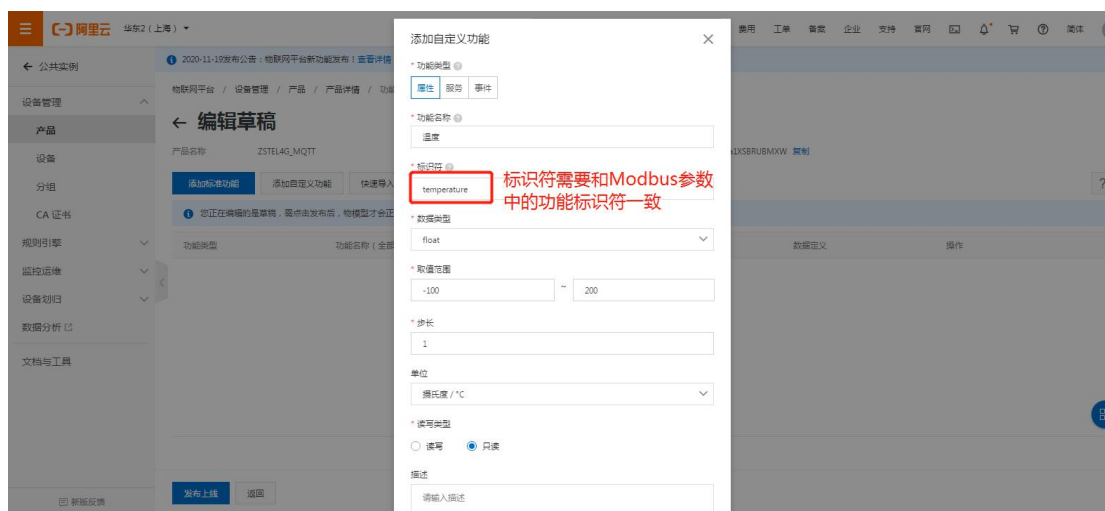


(2) 点击添加自定义功能

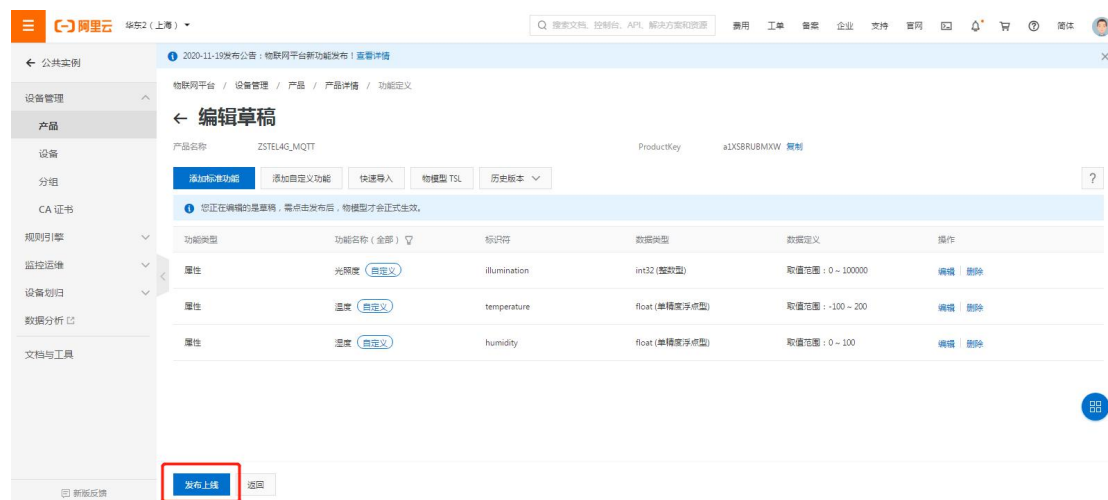


(3) 按照采集数据的类型添加功能。

注：标识符需要和 Modbus RTU 参数中的功能标识符保持一致，这样阿里云 IoT 平台平台才能正确解析物理模型



(4)功能定义完以后点左下角发布



3.1.10 当设备采集数据并上报到阿里云 IoT 平台, 可以在设备-物理模型数据查看。现在就实现了 Modbus RTU 转 json 并上报到阿里云 IoT 平台的功能。



3.1.11 阿里云下发设置

如果配置了可读可写的开关量类型, 比如继电器, 可以使用阿里云设置功能。先在阿里云 IoT 平台物理模型添加一个继电器。

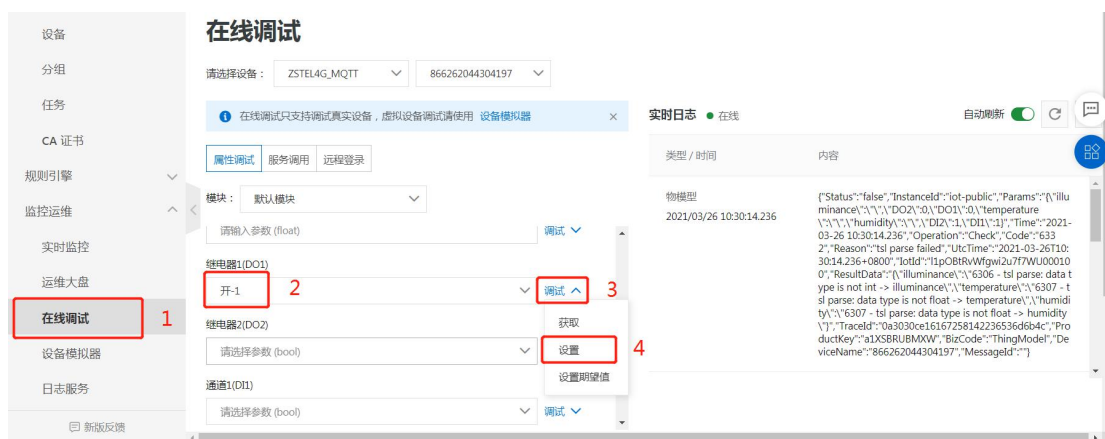


3.1.12 在 Modbus 参数配置软件配置 IO 口的 Modbus RTU 相关参数 (以众山 RTU 为例, 默认 Modbus RTU 地址为 100, DI1、DI2 的寄存器地址分别为 17、18; DO1、DO2 的寄存器地址分别为 20、21, 数据类型为 bool)



#	功能标识符	描述	数据区	Modbus子站地址	Modbus寄存器地址	采集数据类型	字节序	公式
0	humidity	湿度	4X(Holding Status)	1	0	uint16	1234	value=value...
1	temperature	温度	4X(Holding Status)	1	1	int16	1234	value=value...
2	illuminance	光照度	4X(Holding Status)	1	7	uint32	1234	
3	DI1	通道1	4X(Holding Status)	100	16	bool	1234	
4	DI2	通道2	4X(Holding Status)	100	17	bool	1234	
5	DO1	继电器1	4X(Holding Status)	100	20	bool	1234	
6	DO2	继电器2	4X(Holding Status)	100	21	bool	1234	

3.1.13 从阿里云 IoT 平台的在线调试功能下发控制继电器的指令



阿里云 IoT 平台下发的原始指令为 json 格式, 网关 DTU 会转换为相应的 Modbus 指令控制继电器, RTU 返回的 Modbus 指令也会打包成 json 然后立即上报, 这样可以实时控制和获取继电器状态。

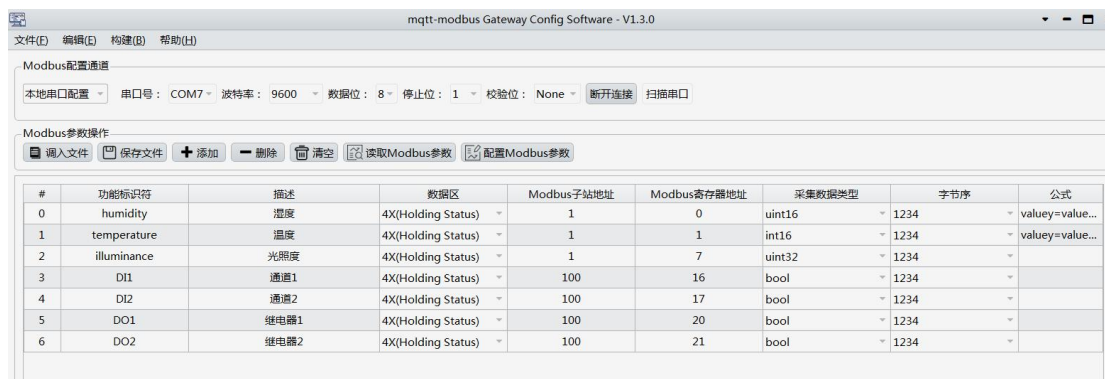
3.2 自建服务器-MQTT 协议

3.2.1 参数配置

配置方式和阿里云大致相同，网络协议选择 MQTT，配置服务器的 IP/域名和端口号，MQTT 的 client ID、username、password 发布主题和订阅主题，json 格式选择自建服务器。



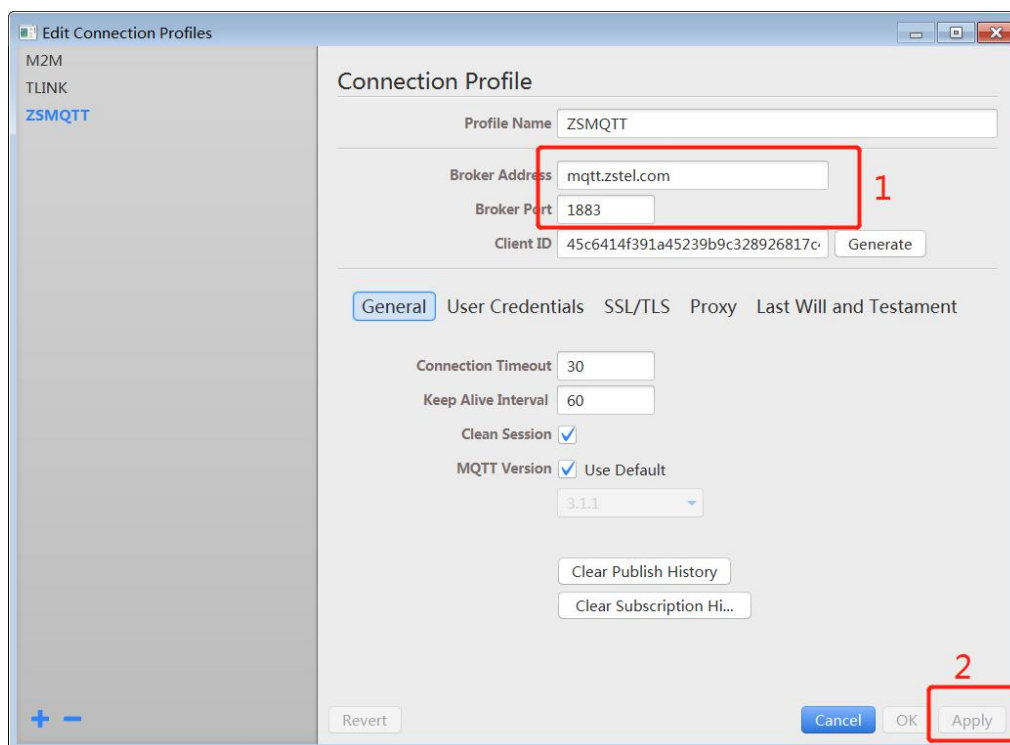
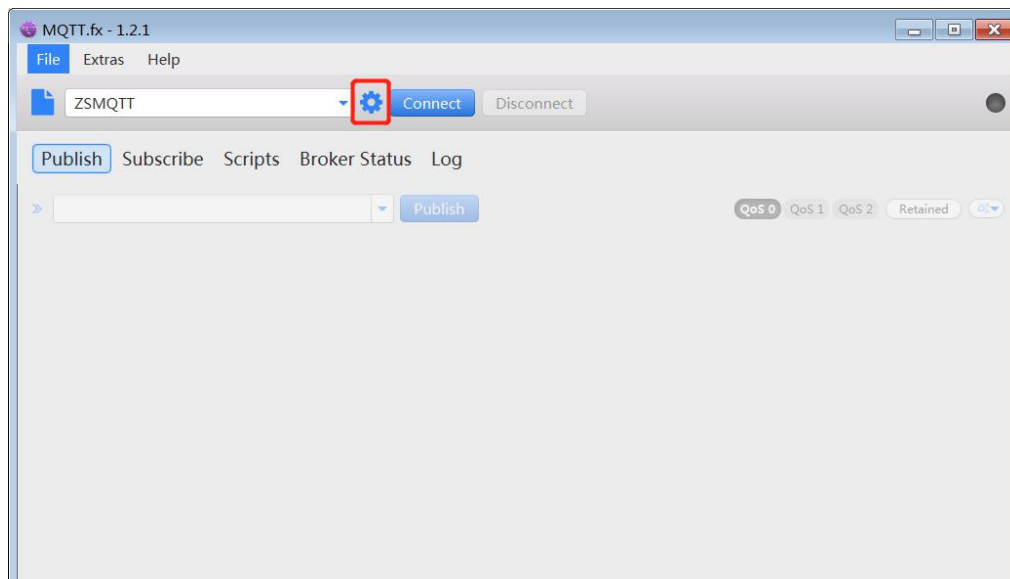
Modbus RTU 参数配置如下：（具体说明见上一章阿里云 IoT 平台 Modbus RTU 参数配置）



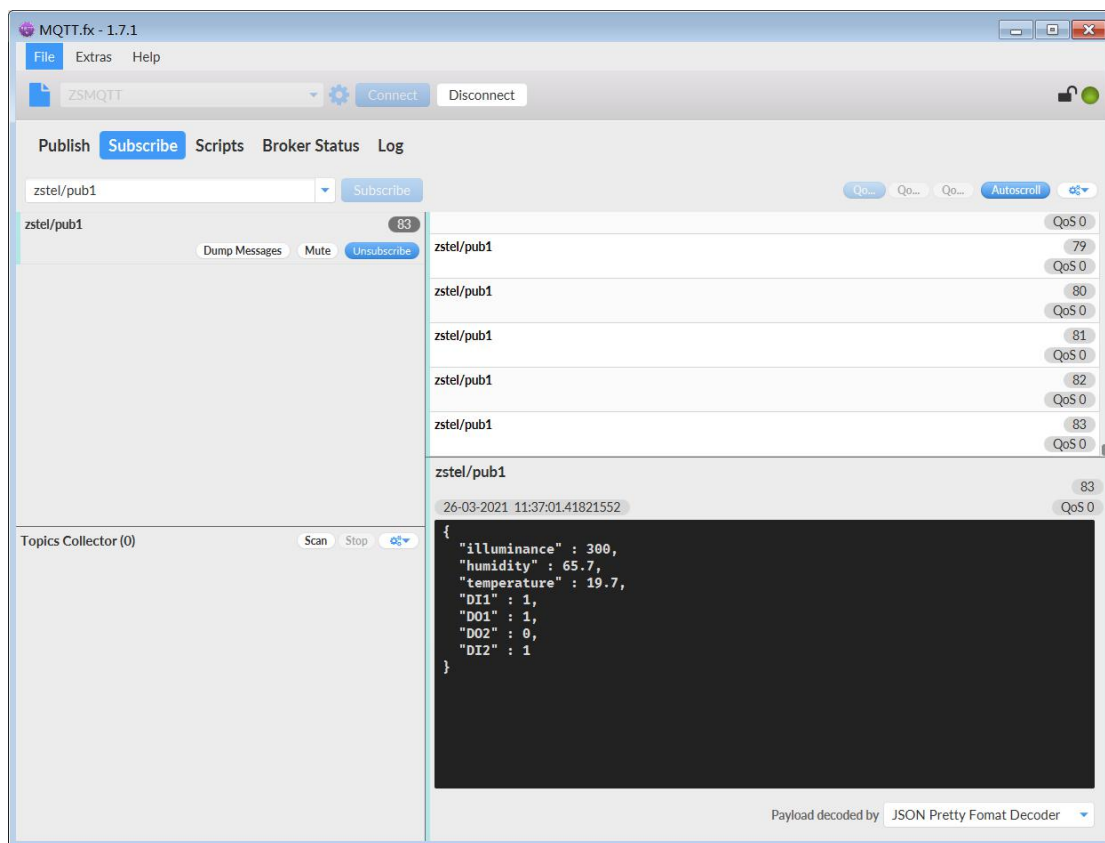
参数配置成功以后需要点复位设备，参数才会生效

3.2.2 下载 MQTT.fx 软件，配置连接服务器相关参数

(1) 点设置按钮，配置 MQTT 服务器 IP 和端口



(2) 连接成功以后配置订阅号，设置为 DTU 参数的发布号，然后点击 Subscribe 订阅，当 DTU 采集数据并上传以后就可以下面的消息框查看数据了，数据为 json 格式，内容为解析后的实际值。



上报的 json 格式为：

```
{
  "illuminance" : 300,
  "humidity" : 65.7,
  "temperature" : 19.7,
  "DI1" : 1,
  "DO1" : 1,
  "DO2" : 0,
  "DI2" : 1
}
```

"illumination" : 300 表示光照度为 310Lux

"humidity" : 65.7 表示湿度为 65.7%

"temperature" : 19.7 表示温度为 19℃

"DI1" : 1 表示 DI1 为高电平（断）

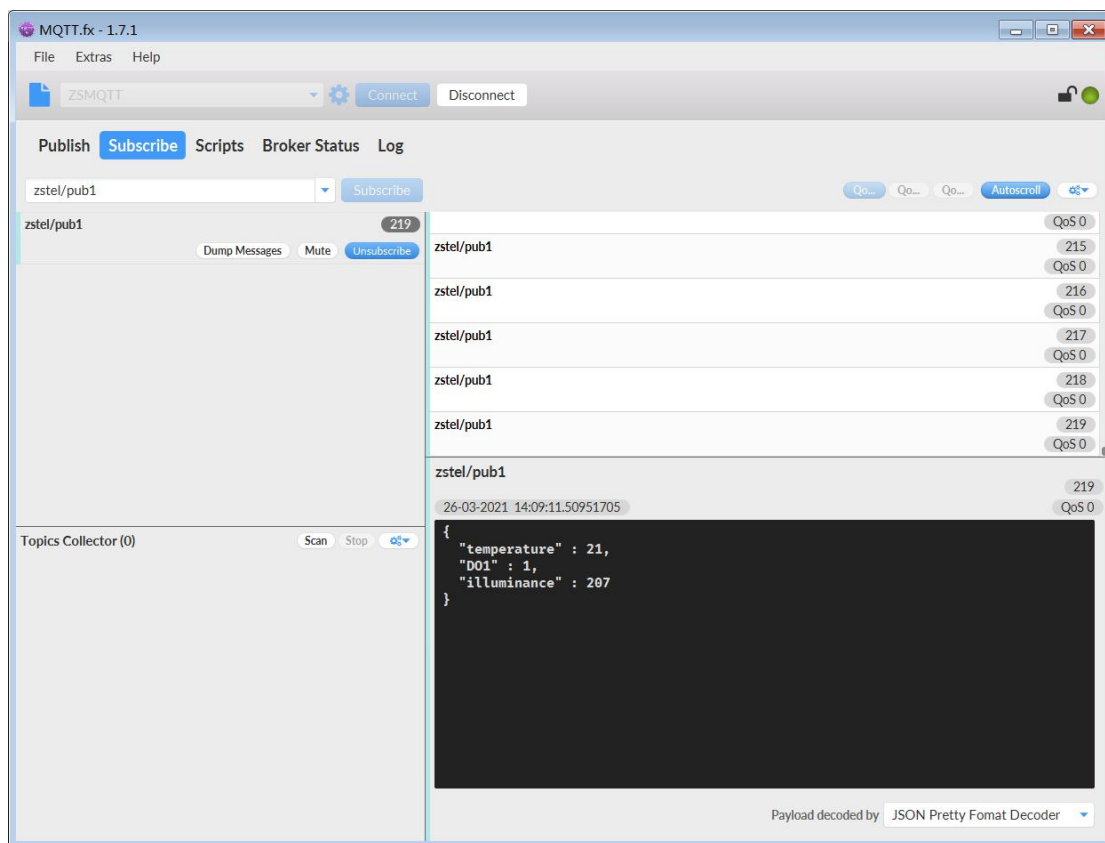
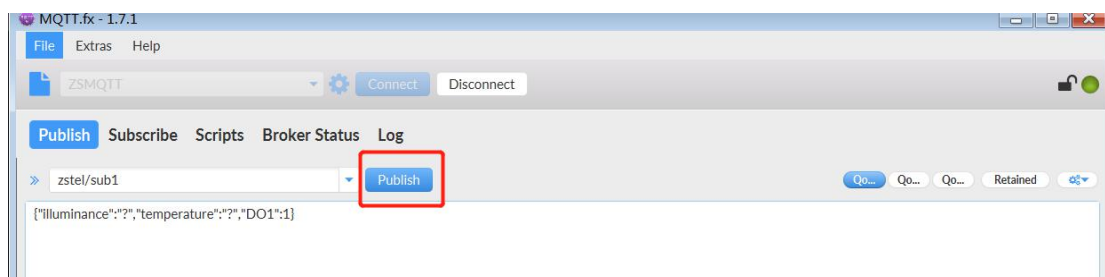
"DO1" : 1 表示 DO1 为高电平（开）

"DO2" : 0 表示 DO2 为低电平（关）

"DI2" : 1 表示 DI2 为高电平（断）

3.2.3 json 指令下发

网关版本 4G DTU 支持下发 json 格式数据转 Modbus RTU 指令，读取指令格式为 {"key1": "?", "key2": "?", ...}; 设置指令格式为 {"key1": 0, "key2": 100, ...}。可以读取/设置一个 key 或多个 key，也可以同时读取和设置 key，比如 {"key1": "?", "key2": 1, ...}，key 必须为已经在 Modbus 参数里配置好的功能标识符名称，或者是系统关键字（第六章有详细说明），对顺序没要求。响应的 Modbus RTU 指令或者系统关键字参数也会立即打包成 key-value 的 json 格式并上报，value 值为获取的实时数据或者配置的数据。



3.3 自建服务器-TCP 协议

3.3.1 参数配置

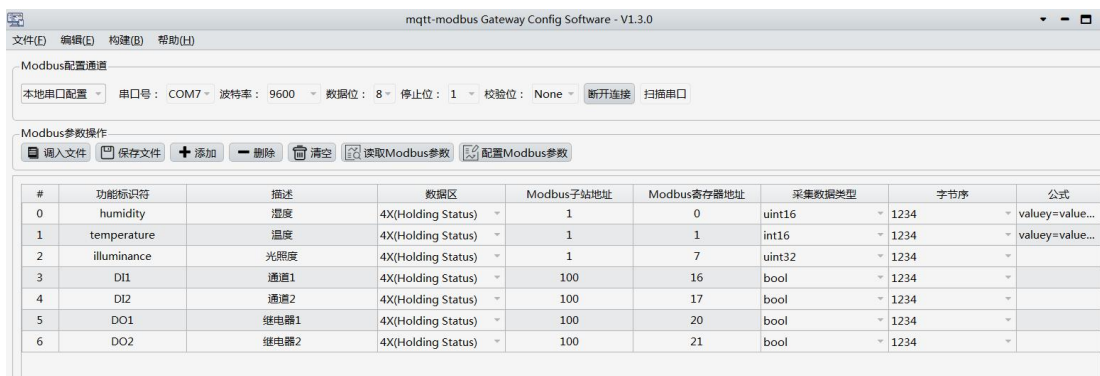
网络协议选择 TCP-Client，配置服务器 IP 地址/域名和端口号，json 格式选择自建服务器。

<4G-自建数据中心1参...		
中心1网络通信协议	TCP-Client	TCP/UDP/HTTP/MQTT通信协议选择
主数据中心1 IP地址或...	mqtt.zstel.com	主数据中心1的IP地址或域名
主数据中心1 侦听端口号	16001	建议使用1024-65000之间的端口号
备用数据中心1 IP地址...		仅TCP/UDP协议支持备用中心
备用数据中心1 侦听端...		建议使用1024-65000之间的端口号
json格式	自建数据中心	上传到不同的服务器可能需要不同的json格式,打开json转换功能有效

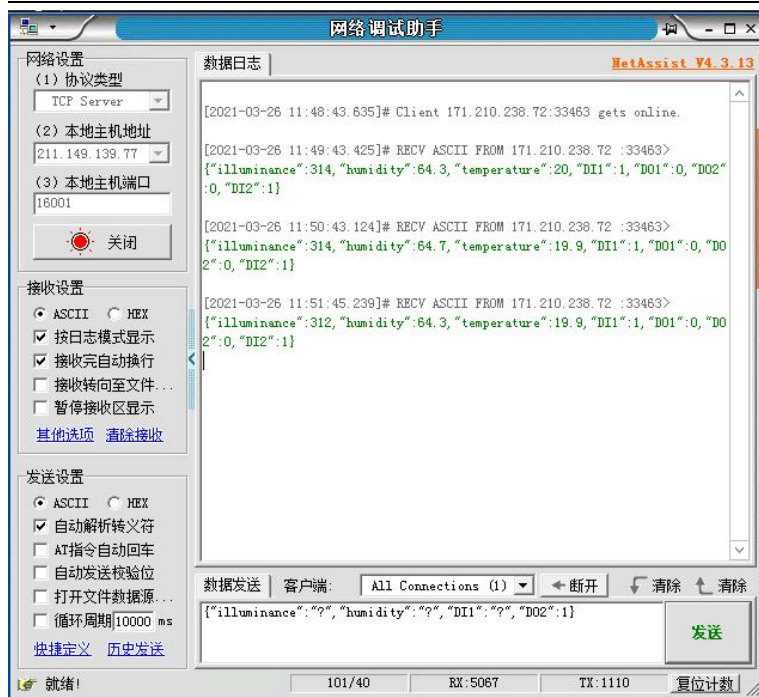
配置变化周期和上报周期

<4G-JSON相关参数>			
00A7	Modbus转json开关	开启	打开使用Modbus转JSON功能,关闭使用透传功能
00B9	固定上传周期	60000	将json数据上传到服务器的时间间隔,单位为毫秒
00B8	变化周期	10000	采集每个标识符的时间间隔,单位为毫秒
00A0	用户json模板		用户自定义json模板,必须带value:{}用于替换原始json

Modbus RTU 参数配置如下：（具体说明见 3.1 阿里云 IoT 平台 Modbus RTU 参数配置）



3.3.2 配置好参数以后复位设备，在服务器上用 socket 软件打开端口，就可以接收到 DTU 上报的 json 数据。



3.3.3 json 指令下发

参考 3.2.3 节，下发指令格式一致

注：除了 MQTT/TCP-Client 协议，UDP-Master/UDP-ZSD/TCP-ZSD/HTTP 协议也都支持

Modbus 转 json 和 json 下发读取/设置指令，使用方法和 3.2.3 节一致，只是组网方式不同。

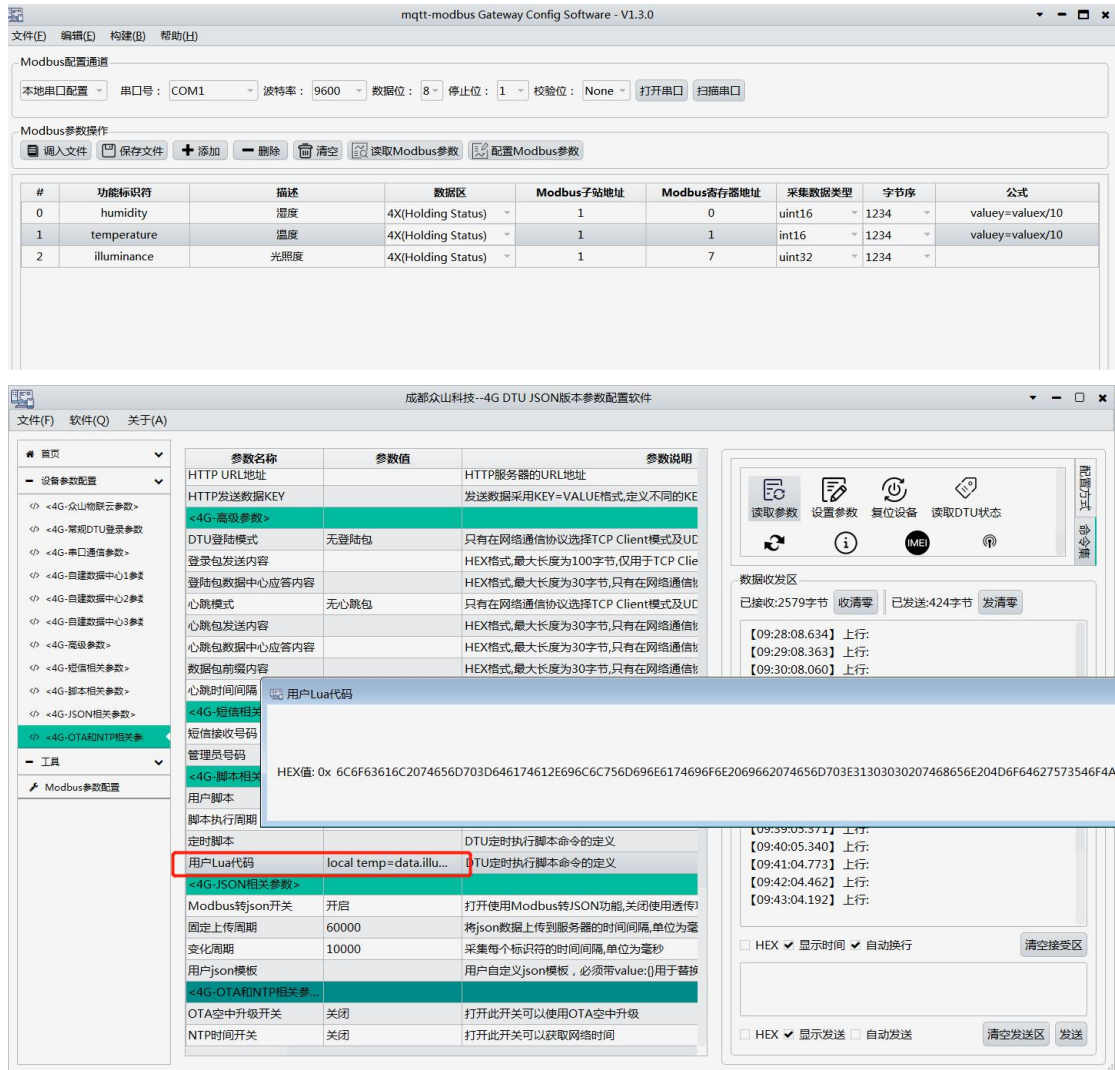
具体组网方式请看《LTE-XXX 4G 网关 DTU 用户手册》。



四、Lua 代码说明

用户可以配置 Lua 代码来判断获取数据的大小，并根据上下限阈值执行不同的操作。

示例：采集一个温湿度、光照度传感器数据，按照 2.2 节说明配置好相应的参数



配置的 Lua 代码：

```
local temp=data.illumination if temp>1000 then ModbusTojson.count=ModbusTojson.count+1
count_str=tostring(ModbusTojson.count) SHELL="@CS=count:~count_str if
ModbusTojson.count==5 then ModbusTojson.count=0 SHELL="@CS=High illumination
warning!@D=1@DO1=1" end else SHELL="@CS=illumination
normal@D=1@DO1=0@PUB=1" end
```

注：如果Lua代码太长，配置的对话框也很长，可以直接点Enter回车配置

正常模式Lua代码为(方便阅读)：

```
local temp=data.illumination
if temp>1000 then
    ModbusToJson.count=ModbusToJson.count+1
    count_str=tostring(ModbusToJson.count)
    SHELL="@CS=count: "..count_str
    if ModbusToJson.count==5 then
        ModbusToJson.count=0
        SHELL="@CS=High illumination warning!@D=1@DO1=1@PUB=1"
    end
else
    SHELL="@CS=illumination normal@D=1@DO1=0"
end
```

Lua 代码实现效果：

当光照度小于或等于 1000 时，串口打印 illumination normal，关闭 DO1，并且不往服务器上上报数据；当光照度大于 1000 时，先判断次数，串口打印 count 数量，如果 Lua 脚本连续执行 5 次光照度都大于 1000，串口打印 High illumination warning!，DO1 打开，并且将采集的数据立即上报到服务器。

赋值语句为 local y=data.x，其中 x 为功能标识符（必须为英文），y 为获取的标识符值，后面就是判断变量的大小和执行相应的操作。所有操作可以通过 SHELL 脚本执行，具体请参考脚本编程手册。ModbusToJson.count 表示计数，如果采集很频繁告警以后就会一直上报数据，通过 ModbusToJson.count 计数可以设置上传周期。

说明：

- ①data.x 为获取功能标识符的值，标识符只能配置为英文
- ②count 为计数的个数，初始为 0，可以通过判断 count 数量设置告警周期
- ③@PUB 指令表示是否立即发布数据到服务器，@PUB=1 表示发送，@PUB=0 或者不配置表示不发送，只有等到了上报周期才发送。
- ④“SHELL=”后面可以是所有支持的脚本指令，具体请参考《LTE-xxx 脚本编程手册》
- ⑤当最后一条功能标识符的指令发送完成执行一次 Lua 代码，执行周期=变化周期*标识符

数量

一般流程为：先定义一个变量获取标识符值，设定变量阈值，判断大小，（bool 值则判断真假，0 为假，1 为真），然后根据判断执行具体的操作，所有操作可以通过 SHELL 脚本执行，比如打印信息，上报数据，开/关继电器等。

五、NTP 时间和 GPS 数据获取

5.1 NTP 时间获取

当把功能标识符设置为 TS 时，表示获取 NTP 时间，其他 Modbus RTU 相关参数可以不用设置（无效），上报的 json 包里面会获取 NTP 时间，格式为年/月/日，星期，时：分：秒，比如获取的 TS 时间为 2020/12/02, 03, 14:17:10, 表示 2020 年 12 月 02 日，星期三，14:17:10。

注：如果想使用 NTP 时间功能，需要先把 NTP 时间开关打开

<4G-OTA和NTP相关参...		
OTA空中升级开关	关闭	打开此开关可以使用OTA空中升级
NTP时间开关	开启	打开此开关可以获取网络时间

1) 参数配置

Modbus参数操作								
<input type="button" value="调入文件"/> <input type="button" value="保存文件"/> <input type="button" value="+ 添加"/> <input type="button" value="- 删除"/> <input type="button" value="清空"/> <input type="button" value="恢复Modbus参数"/> <input type="button" value="配置Modbus参数"/>								
#	功能标识符	描述	数据区	Modbus子站地址	Modbus寄存器地址	采集数据类型	字节序	公式
0	humidity	湿度	4X(Holding Status)	1	0	uint16	1234	valuey=valuex/10
1	temperature	温度	4X(Holding Status)	1	1	int16	1234	valuey=valuex/10
2	illuminance	光照度	4X(Holding Status)	1	7	uint32	1234	
3	TS		0X(Coil Status)			int16	1234	

2) TS 时间上报

Publish
Subscribe
Scripts
Broker Status
Log

zstel/pub1
Subscribe
QoS 0
QoS 0
QoS 0
Autoscroll

zstel/pub1
Dump Messages
Mute
Unsubscribe
4
zstel/pub1
QoS 0
zstel/pub1
QoS 0
zstel/pub1
QoS 0
zstel/pub1
QoS 0
zstel/pub1
QoS 0

Topics Collector (0)
Scan
Stop
03-03-2021 10:13:46.36826661
QoS 0

```

{
  "humidity": 56.9,
  "temperature": 17.2,
  "TS": "2021/03/03,03,10:15:01",
  "illuminance": 319
}

```

Payload decoded by
JSON Pretty Format Decoder

5.2 GPS 数据采集

注：GPS 版本才支持此指令，否则指令无效

5.2.1 采集指令

如果是带 GPS 版本，可以将功能标识符配置为 GPS_xx 来获取 GPS 数据，xx 可以是以下内容，不同的指令获取的 GPS 格式也不同。具体见下表：

指令	名称	格式	含义
GPS_xx	GPS 定位信息 采集	GPS_GGA	采集标准的 GGA 信息
		GPS_RMC	采集标准的 RMC 信息
		GPS_GSA	采集标准的 GSA 信息
		GPS_JW	只采集经纬度，格式为 经度,纬度
		GPS_WJ	只采集经纬度，格式为 纬度,经度

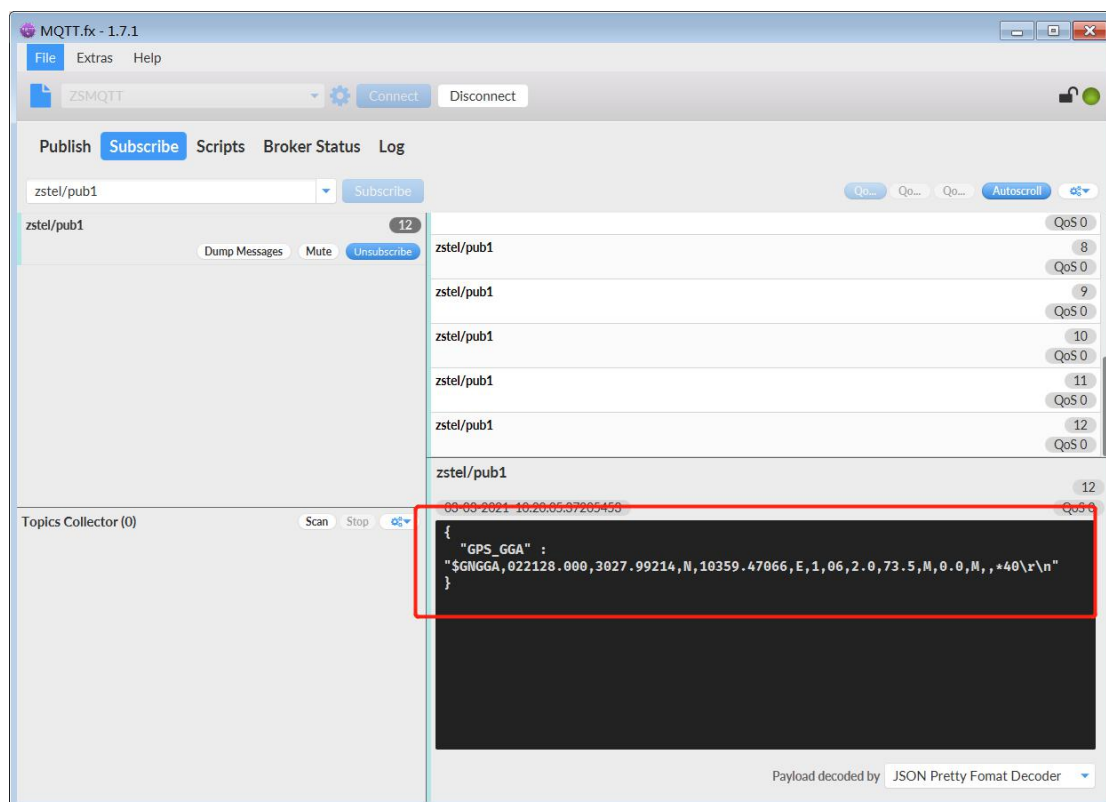
参数配置：

只需要将功能标识符配置为 GPS_GGA，其他参数不用配置。



#	功能标识符	描述	数据区	Modbus子站地址	Modbus寄存器地址	采集数据类型	字节序	公式
0	GGA	0X(Coil Status)	0X(Coil Status)			int16	1234	

服务器接收的数据如下：



5.2.2 报文解析

示例：

1) 配置标识符为 GPS_GGA，采集的 GPS 数据为标准的 GGA 数据，格式解析如下：

\$GPGGA,<1>,<2>,<3>,<4>,<5>,<6>,<7>,<8>,<9>,M,<10>,M,<11>,<12>*xx

<1> UTC 时间，格式为 hhmmss.sss；

<2> 纬度，格式为 ddmm.mmmm(第一位是零也将传送)；

<3> 纬度半球，N 或 S(北纬或南纬)

<4> 经度，格式为 dddmm.mmmm(第一位零也将传送)；

<5> 经度半球，E 或 W(东经或西经)

<6> 定位质量指示，0=定位无效，1=定位有效；

<7>使用卫星数量，从 00 到 12(第一个零也将传送)

<8>水平精确度，0.5 到 99.9

<9 天线离海平面的高度, -9999.9 到 9999.9 米 M 指单位米

<10>大地水准面高度, -9999.9 到 9999.9 米 M 指单位米

<11>差分 GPS 数据期限(RTCMSC-104), 最后设立 RTCM 传送的秒数量

<12>差分参考基站标号, 从 0000 到 1023(首位 0 也将传送)。

2) 配置标识符为 GPS_JW, 则只返回经纬度信息, 比如 104.10194,30.65984。104.10194 为经度, 30.65984 为纬度, 可以用此经纬度在地图上定位。

六、用户自定义 json 模板

用户可以自定义 json 格式，通过用户 json 模板参数可以在 json 包里添加一些自定义标识符，比如 DTU ID，Device name，Location，ts 时间戳等信息。自定义 json 内容可以直接添加到原始 json 包内，也可以添加到原始的 json 包外，或者添加多重 json。

<4G-JSON相关参数>		
Modbus转json开关		打开使用Modbus转JSON功能,关闭使用透传功能
固定上传周期		将json数据上传到服务器的时间间隔,单位为毫秒
变化周期		采集每个标识符的时间间隔,单位为毫秒
用户json模板		用户自定义json模板, 必须带value:{}用于替换原始json
<4G-OTA和NTP相关参数>		

注: "ts": "?1" 返回 UNIX 格式的时间戳; "ts": "?2" 返回年/月/日,星期,时:分:秒格式的时间戳

采集温湿度、光照度数据，原始 json 包为:

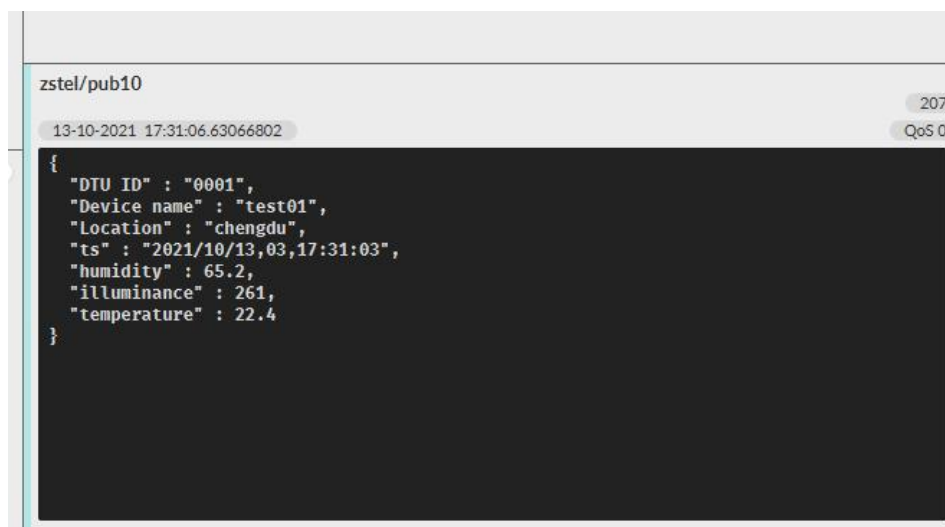
```
{
  "illuminance" : 300,
  "humidity" : 65.7,
  "temperature" : 19.7
}
```

例 1: 添加到原始的 json 包内

配置 json 模板为:

```
{"DTU ID": "0001", "Device name": "test01", "Location": "chengdu", "ts": "?2"}
```

则上报到服务器的 json 包为:

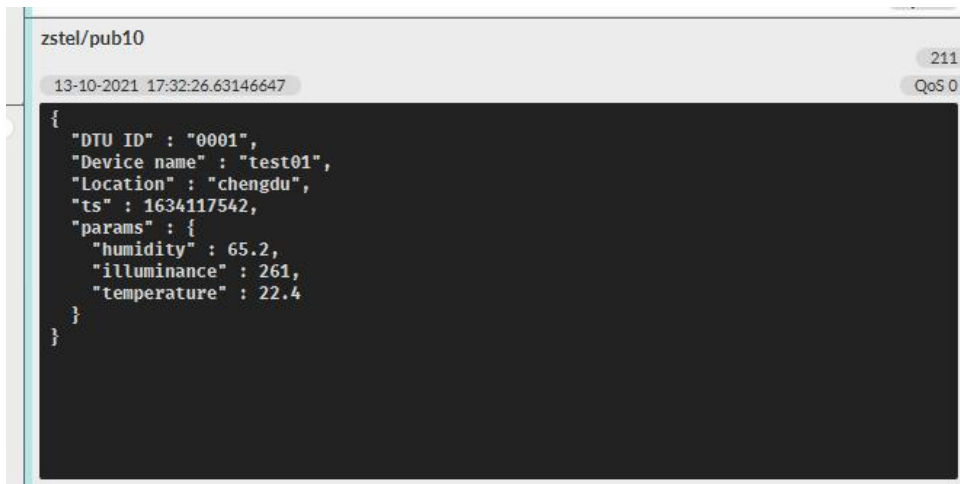


例 2：添加到原始的 json 包外

配置 json 模板为：

```
{"DTU ID":"0001","Device name":"test01","Location":"chengdu","ts":"?1","params":{}}
```

则上报到服务器的 json 包为：



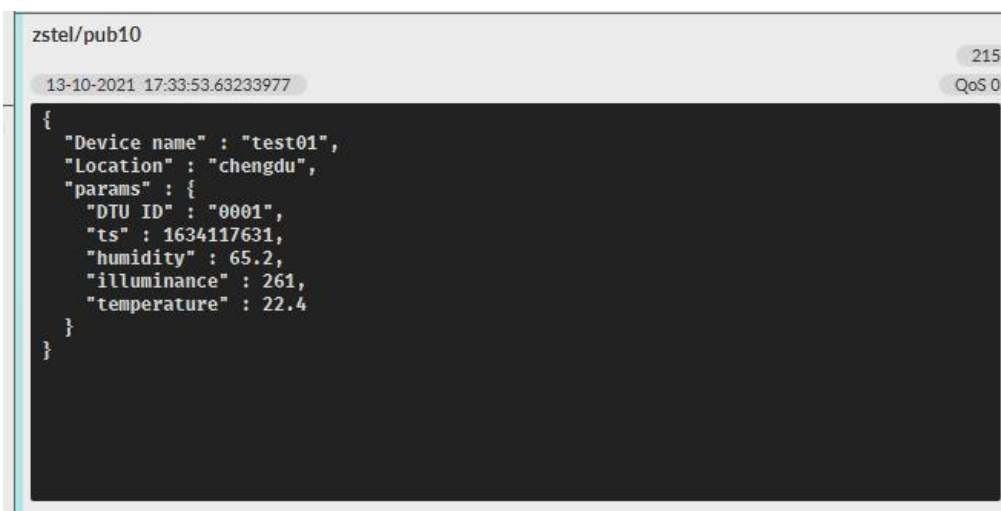
```
zstel/pub10
13-10-2021 17:32:26.63146647 211 QoS 0
{
  "DTU ID" : "0001",
  "Device name" : "test01",
  "Location" : "chengdu",
  "ts" : 1634117542,
  "params" : {
    "humidity" : 65.2,
    "illuminance" : 261,
    "temperature" : 22.4
  }
}
```

例 3：添加到原始的 json 包内部和外部

配置 json 模板为：

```
{"Device name":"test01","Location":"chengdu","params":{"DTU ID":"0001","ts":"?1"}}
```

则上报到服务器的 json 包为：



```
zstel/pub10
13-10-2021 17:33:53.63233977 215 QoS 0
{
  "Device name" : "test01",
  "Location" : "chengdu",
  "params" : {
    "DTU ID" : "0001",
    "ts" : 1634117631,
    "humidity" : 65.2,
    "illuminance" : 261,
    "temperature" : 22.4
  }
}
```

例 4：添加多重 json 包

```
{"Devicename":"test01","DTU ID":"0001","Location":"chengdu",
"params":[{"property":"?1","value":"?2"}]}
```

注：红色的"?1"和"?2"是固定的格式，不能修改。"?1"表示获取标识符名称，"?2"表示获取标识符对应的值。

```
zstel/pub10
13-10-2021 17:35:33.63333636
220
QoS 0
{
  "Device name" : "test01",
  "DTU ID" : "0001",
  "Location" : "chengdu",
  "params" : [ {
    "property" : "humidity",
    "value" : 65.2
  }, {
    "property" : "temperature",
    "value" : 22.4
  }, {
    "property" : "illuminance",
    "value" : 259
  } ]
}
```

七、其他特殊关键字(服务器下发 json 指令)

7.1 SHELL

可以从服务器下发 json 格式的 SHELL 脚本，DTU 会临时执行一次脚本。

7.1.1 阿里云 IoT 平台下发 SHELL 脚本

(1)在阿里云 IoT 平台定义物理模型功能标识符为 SHELL，数据类型为 text，长度为 1024。

编辑自定义功能
×

* 功能类型
属性

* 功能名称 ?
SHELL脚本

* 标识符 ?
SHELL

* 数据类型
text

* 数据长度:
1024 字节

* 读写类型
☒ 读写 ☐ 只读

描述
请输入描述
0/100

确认 取消

(2)切换到在线调试，从标识符“SHELL”下发脚本，就可以实现阿里云 IoT 平台下发 SHELL 脚本，执行完毕后会返回脚本内容。脚本的内容为所有支持的脚本指令，具体请参考《LTE-XXX 4G DTU 脚本编程手册》。

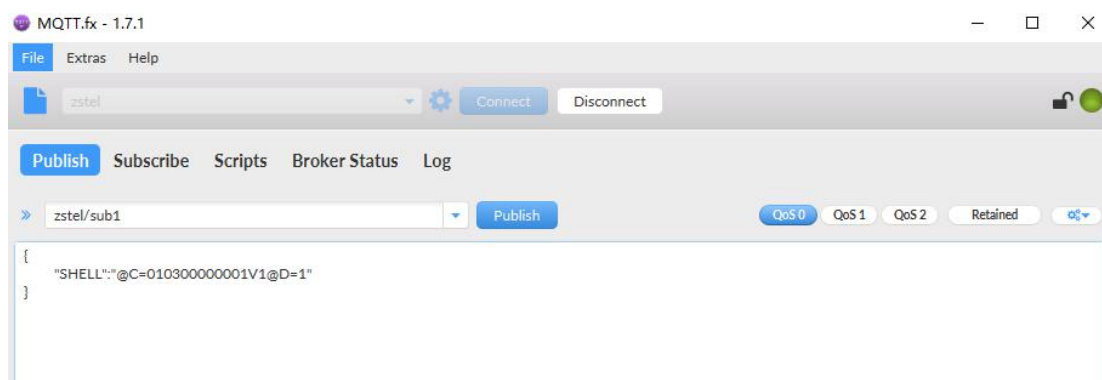
寄存器02(Reg2) ?
 调试 ✓

SHELL脚本(SHELL) ?
 调试 ✓

获取 设置 设置期望值 重置

7.1.2 自建服务器下发 SHELL 脚本

连接上服务器以后，直接从服务器下发{"SHELL":"@C=010300000001V1@D=1"}



7.2 luaCode

可以从服务器下发 json 报文配置用户 Lua 代码。

7.2.1 阿里云 IoT 平台配置用户 Lua 代码

(1)在阿里云 IoT 平台定义物理模型功能标识符为 luaCode，数据类型为 text，长度为 2048。

编辑自定义功能

×

* 功能类型

属性

* 功能名称

用户代码

* 标识符

luaCode

* 数据类型

text

* 数据长度:

2048

字节

* 读写类型

☒ 读写
 ☐ 只读

描述

请输入描述

0/100

确认

取消

(2)切换到在线调试，从标识符“luaCode”下发用户 Lua 代码。例如采集温湿度传感器数据，

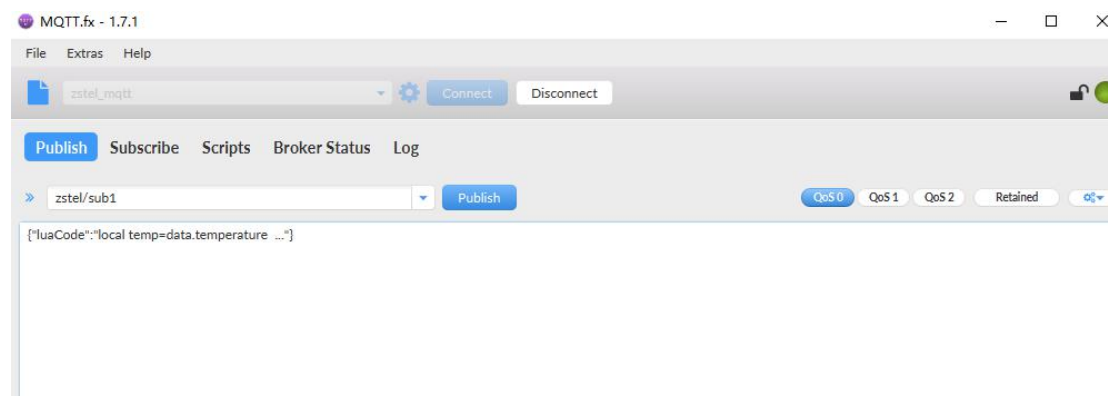
可以在 lua 代码里面获取温度值，然后在做相应的操作，Lua 代码设置成功以后会返回到服

务器。



7.2.2 自建服务器配置用户 Lua 代码

连接上服务器以后，直接从服务器下发{"luaCode":"local temp=data.temperature ...(some thing you do)"}



7.3 Reg1-Reg10

LTE-xxx 网关版本有 10 个可读可写的寄存器供用户调用，可以用做阈值判断。用户可以自

已配置 Reg 参数的值，只能是数字（可以是小数）。

7.3.1 阿里云 IoT 平台配置 Reg

(1)在阿里云 IoT 平台定义物理模型功能标识符为 Reg1~Reg10，数据类型为 int32，取值范围为 0~int32 最大范围，这里定义的 1000。



(3)切换到在线调试，从标识符 “Reg1” 下发设置的值。

在线调试

请选择设备: ZSTEL4G_MQTT 866262044304197

在线调试只支持调试真实设备, 虚拟设备调试请使用 设备模拟器

属性调试 服务调用 远程登录

模块: 默认模块

请选择参数 (bool) 调试

DO2(DO2) 请选择参数 (bool) 调试

时间(TS) 请输入参数 (text) 调试

用户代码(luaCode) 请输入参数 (text) 调试

寄存器01(Reg1) 100 调试

寄存器02(Reg2) 200 调试

SHELL脚本(SHELL) 请输入参数 (text) 调试

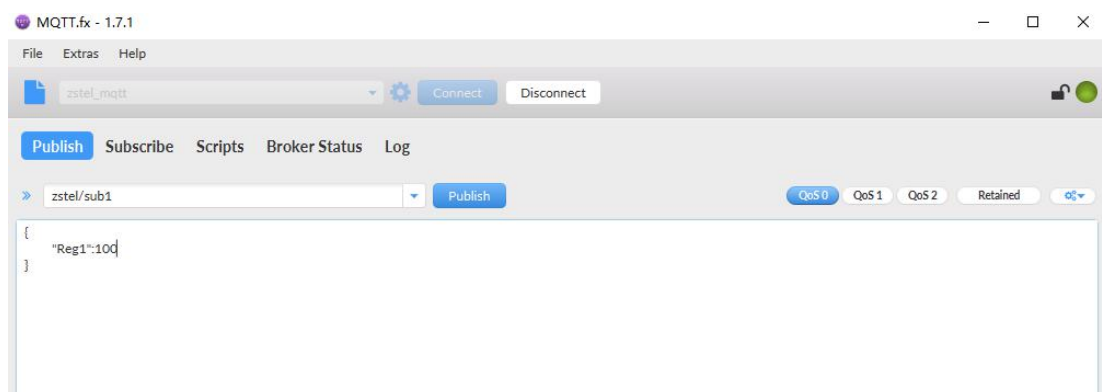
获取 设置 设置期望值 重置

实时日志

时间

7.3.2 自建服务器配置 Reg

连接上服务器以后, 直接从服务器下发{"Reg1":100}



设置成功后, DTU 会立即上报配置的 Reg 值。